QUIC and HTTP/3

Nick Shadrin

HighLoad++ Armenia

Co-organizer

Yandex

About me

- Nick Shadrin

- 20 years in web
  - NetScaler
  - Zscaler

- 8 years at NGINX

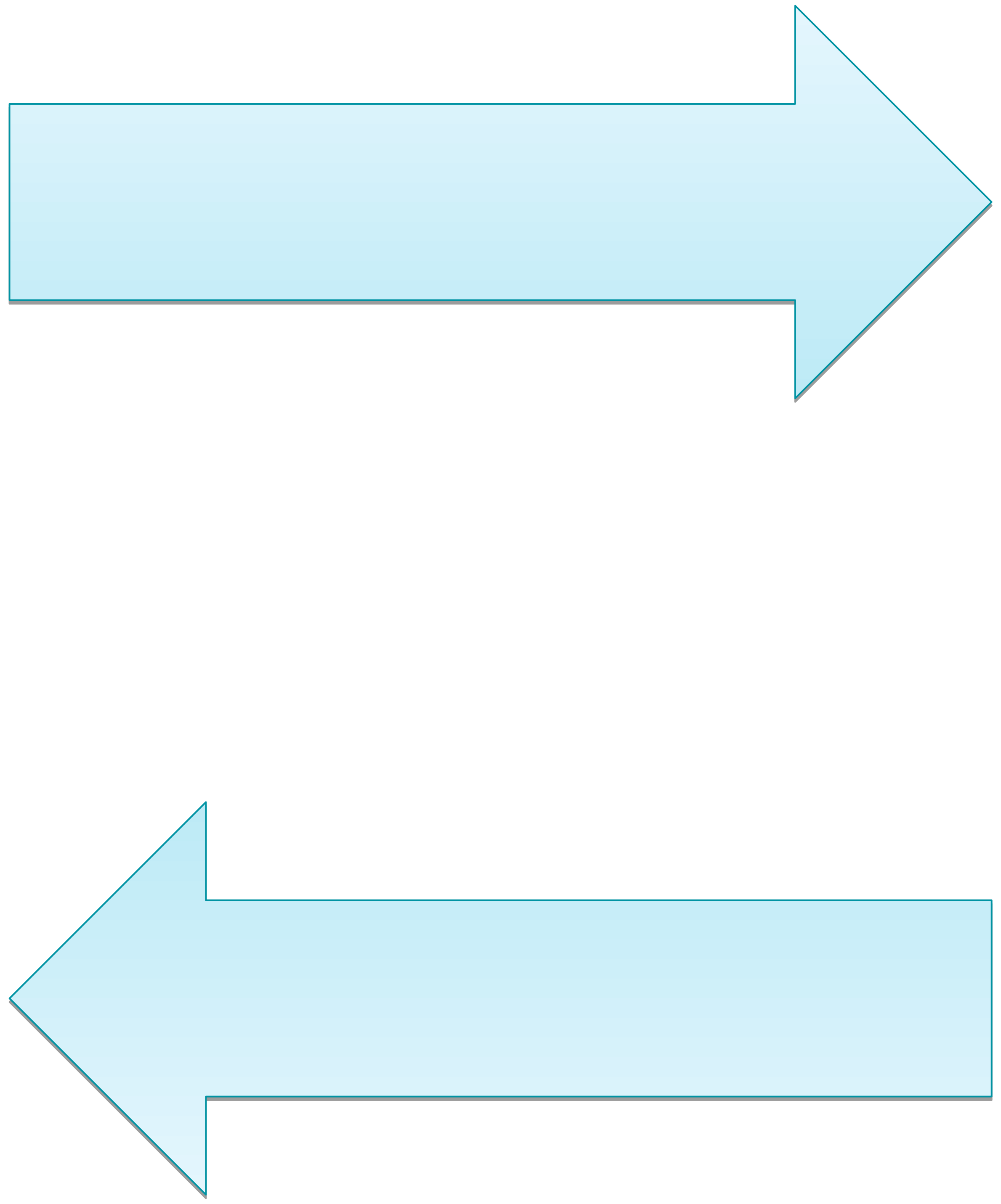
  - Started in Sales Engineering
  - Launched NGINX Unit with Igor and Valentin
  - Now architecting control and management tools

- tg: @nshadrin

HTTP/3 presentation agenda

- History of protocols
  - Main differences
  - Challenge of upgrading to h2
- QUIC and HTTP/3 features
  - UDP
  - Connection ID
  - Encryption
- Real world implementation
- Our favorite part: Q&A

Basics



```
GET /test HTTP/1.1
Host: example.com
User-Agent: Mozilla
X-Forwarded-For: 192.168.10.1
Accept: image/gif, image/jpeg, */*
Accept-Language: en-us
Accept-Encoding: gzip, deflate

HTTP/1.1 301 Moved Permanently
Server: unit/1.9
Date: Thu, 18 Jul 2019 21:19:07 GMT
Content-Type: text/html
Content-Length: 184
Connection: close
Location: https://example.com/test
```

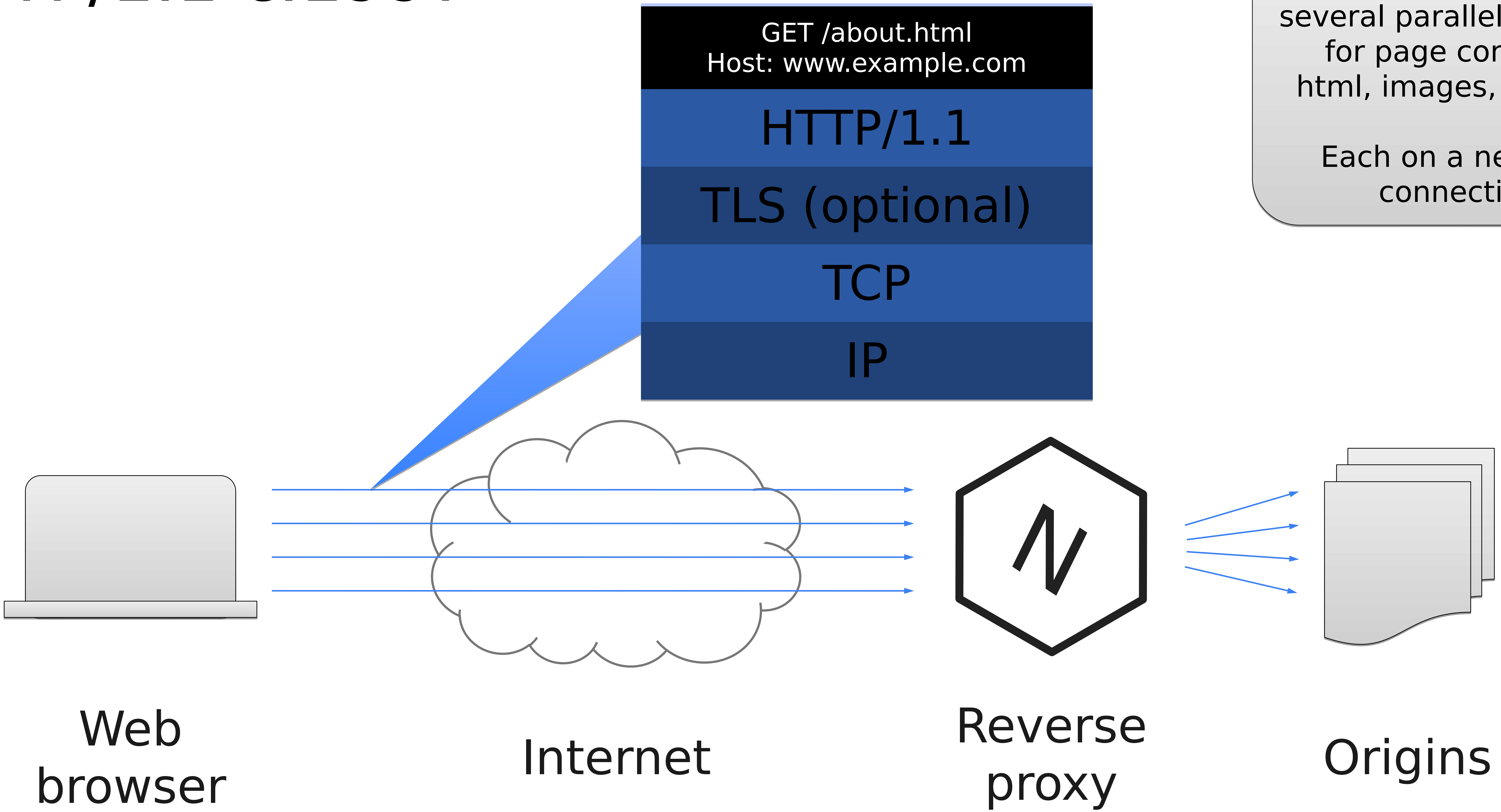
A Brief History of HTTP

Born at CERN
Only GET
Methods
Headers
Cookies!
Keepalive
sessions!
Multiplexing
Server push
HTTP/0.9
HTTP/1.0
HTTP/1.1
HTTP/2.0
HTTP/3.0
1990
1992
1994
1996
1998
2000
2002
2004
2006
2008
2010
2012
2014
2016
2018
2020
2022
QUIC

HTTP/1.1 c.1997

Web browser makes several parallel requests for page contents: html, images, style, JS

Each on a new TCP connection



GET /about.html
Host: www.example.com

HTTP/1.1

TLS (optional)

TCP

IP

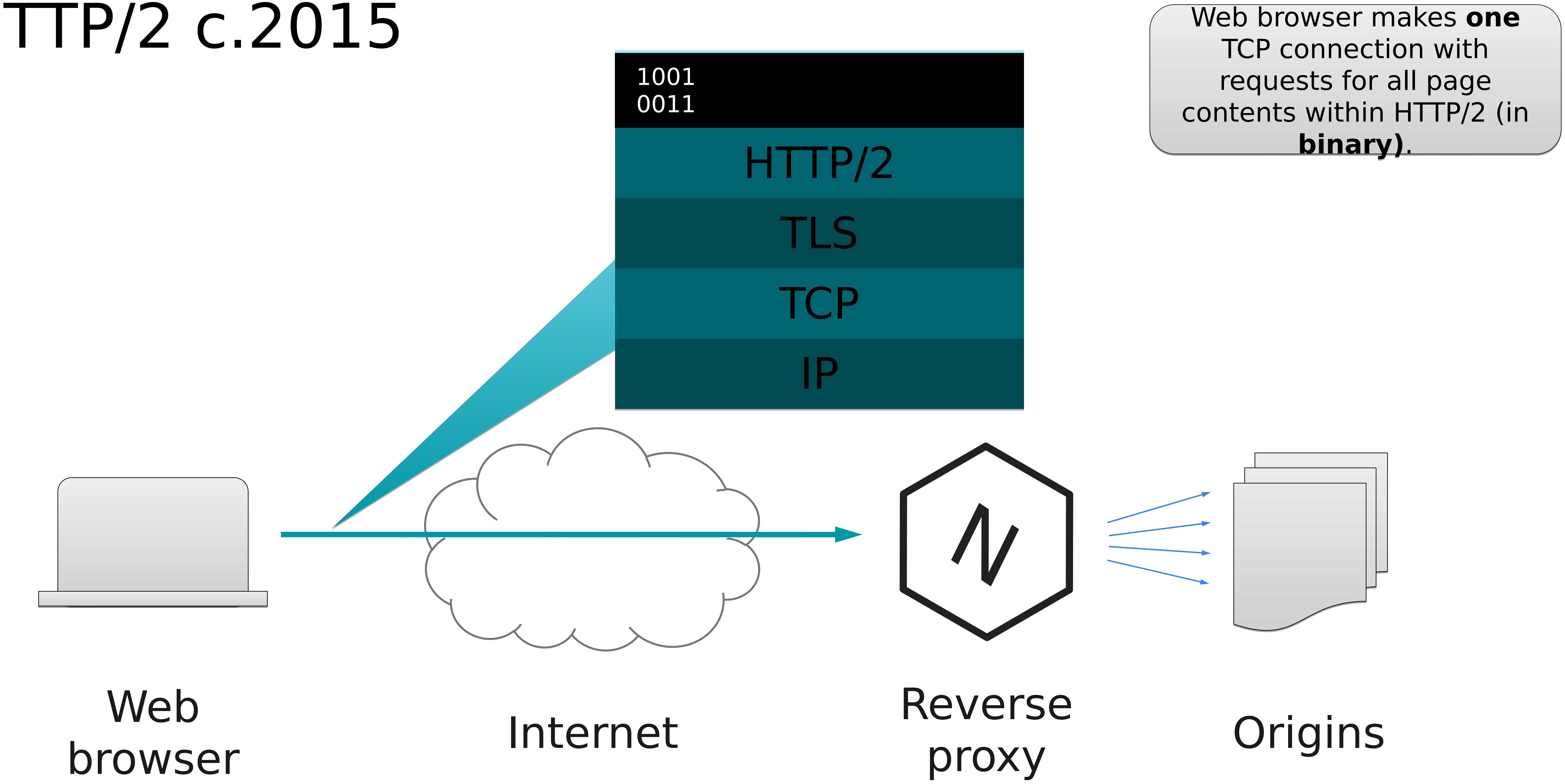
Web browser

Internet

Reverse proxy

Origins

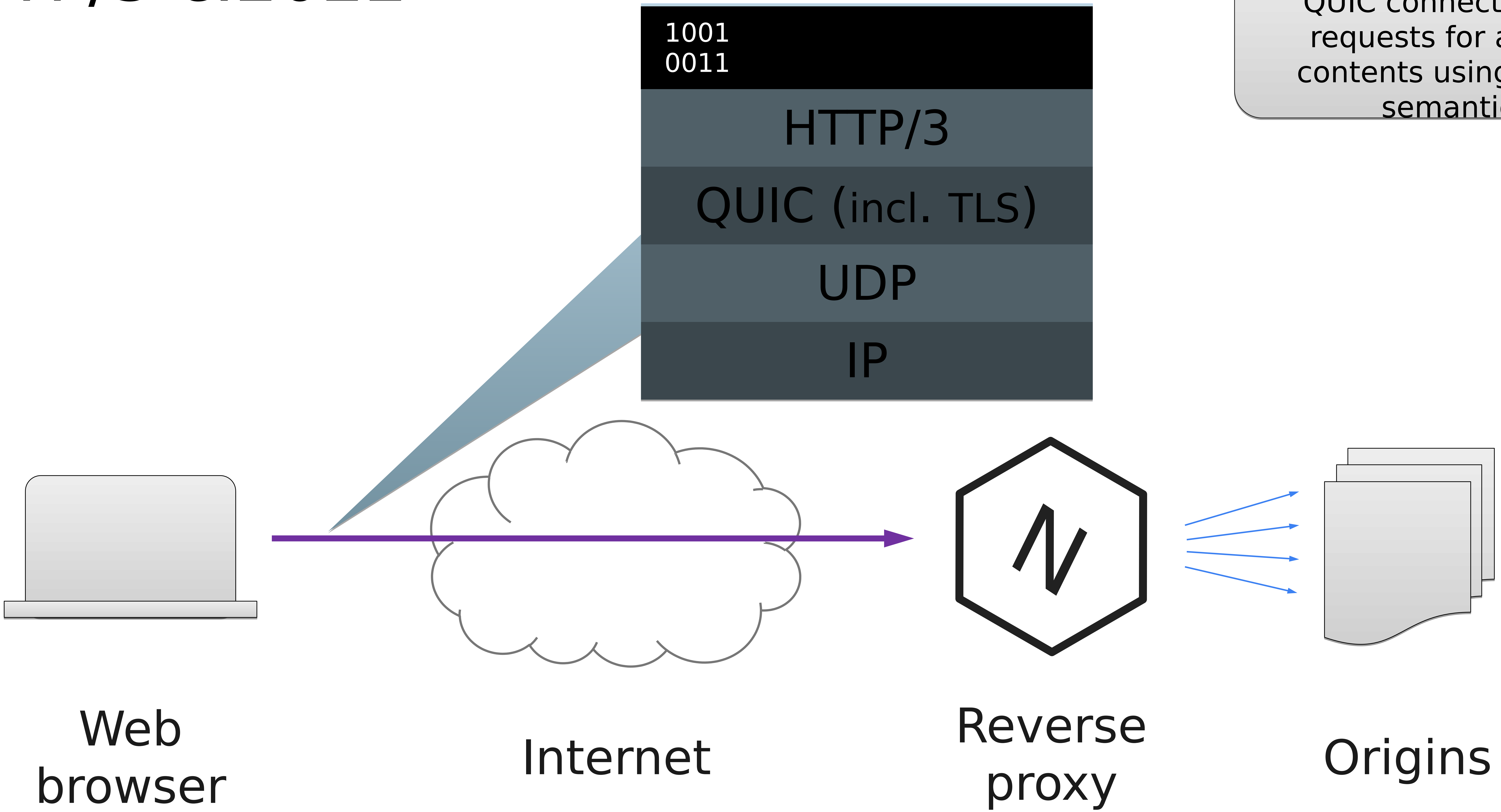
HTTP/2 c.2015



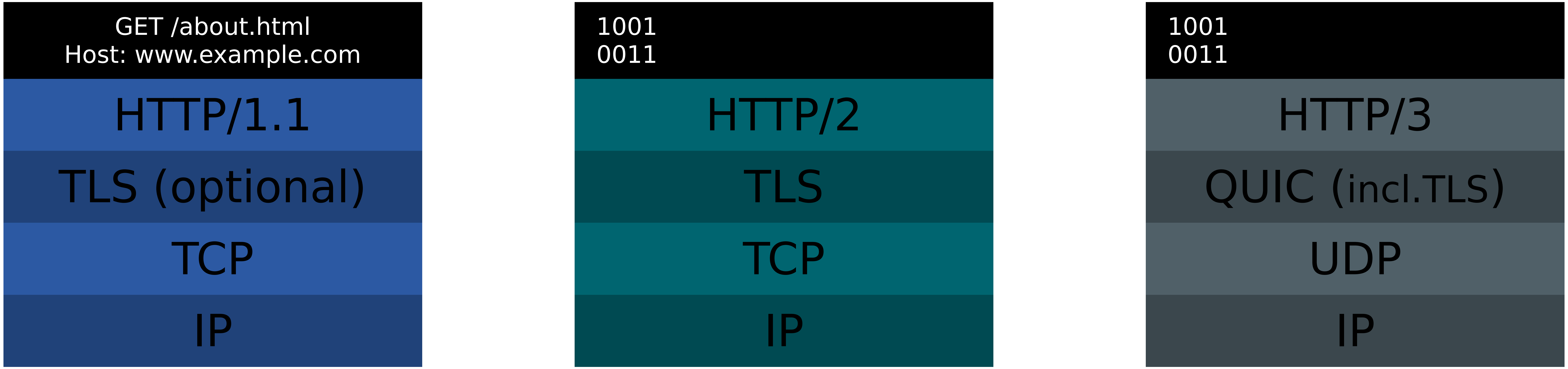
Web browser makes **one** TCP connection with requests for all page contents within HTTP/2 (in **binary)**.

HTTP/3 c.2022

Web browser makes **one** QUIC connection with requests for all page contents using HTTP/3 semantics.



HTTP stacks



GET /about.html
Host: www.example.com

HTTP/1.1

TLS (optional)

TCP

IP

1001
0011

HTTP/2

TLS

TCP

IP

1001
0011

HTTP/3

QUIC (incl.TLS)

UDP

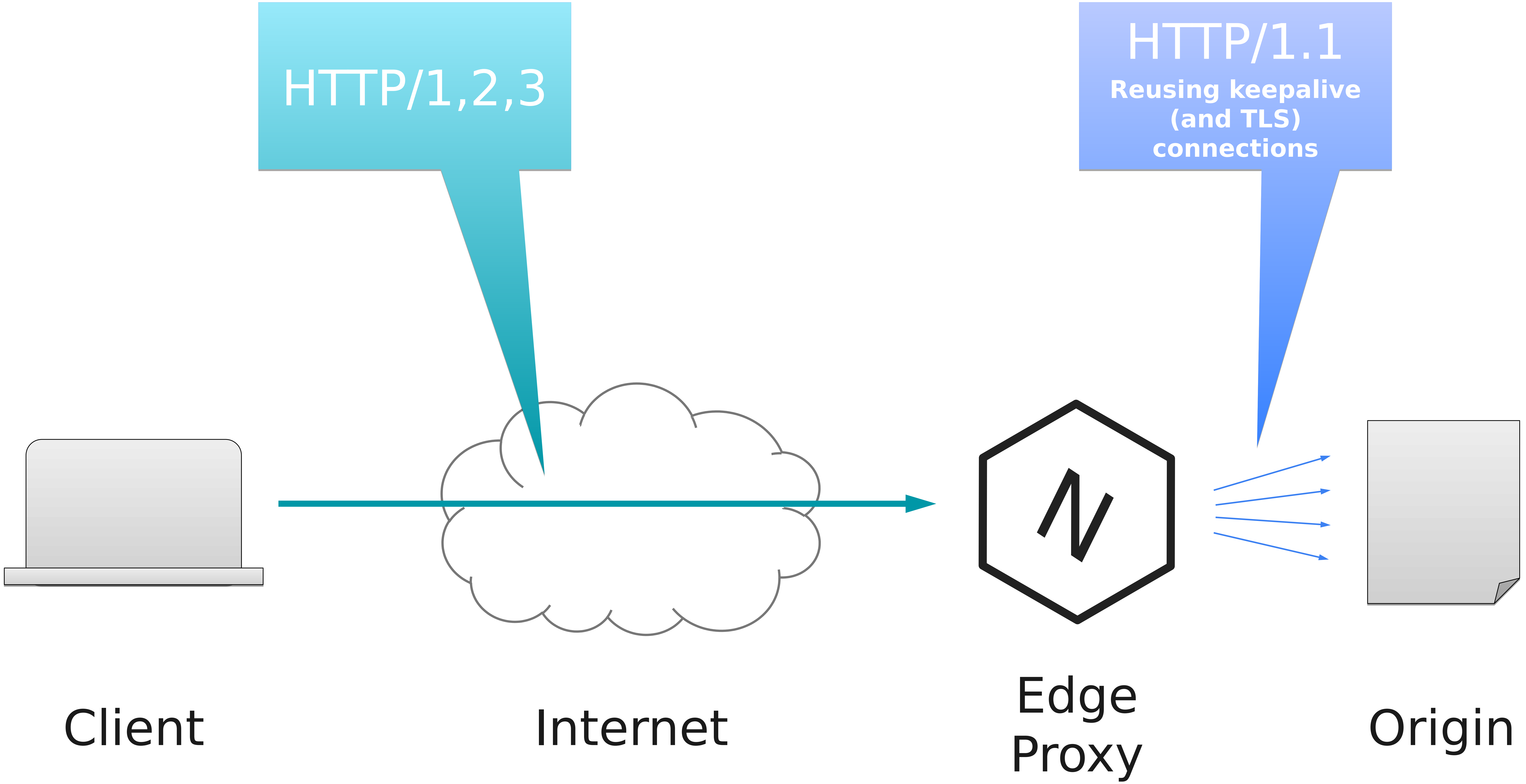
IP

Web browser makes several TCP connections to request page contents:
html, images, style, JS

Web browser makes **one** TCP connection with requests for all page contents in HTTP/2 streams (in **binary)**.

Web browser makes **one** QUIC connection with requests for all page contents using HTTP/3 semantics.

Reality of HTTP deployment



HTTP/1,2,3

HTTP/1.1

Reusing keepalive (and TLS) connections

Don't turn off HTTP/1!

Benefits

- Less reliance on kernel
- Built-in encryption
- Connection ID: migrate connections
- Faster negotiation*

Negotiation history

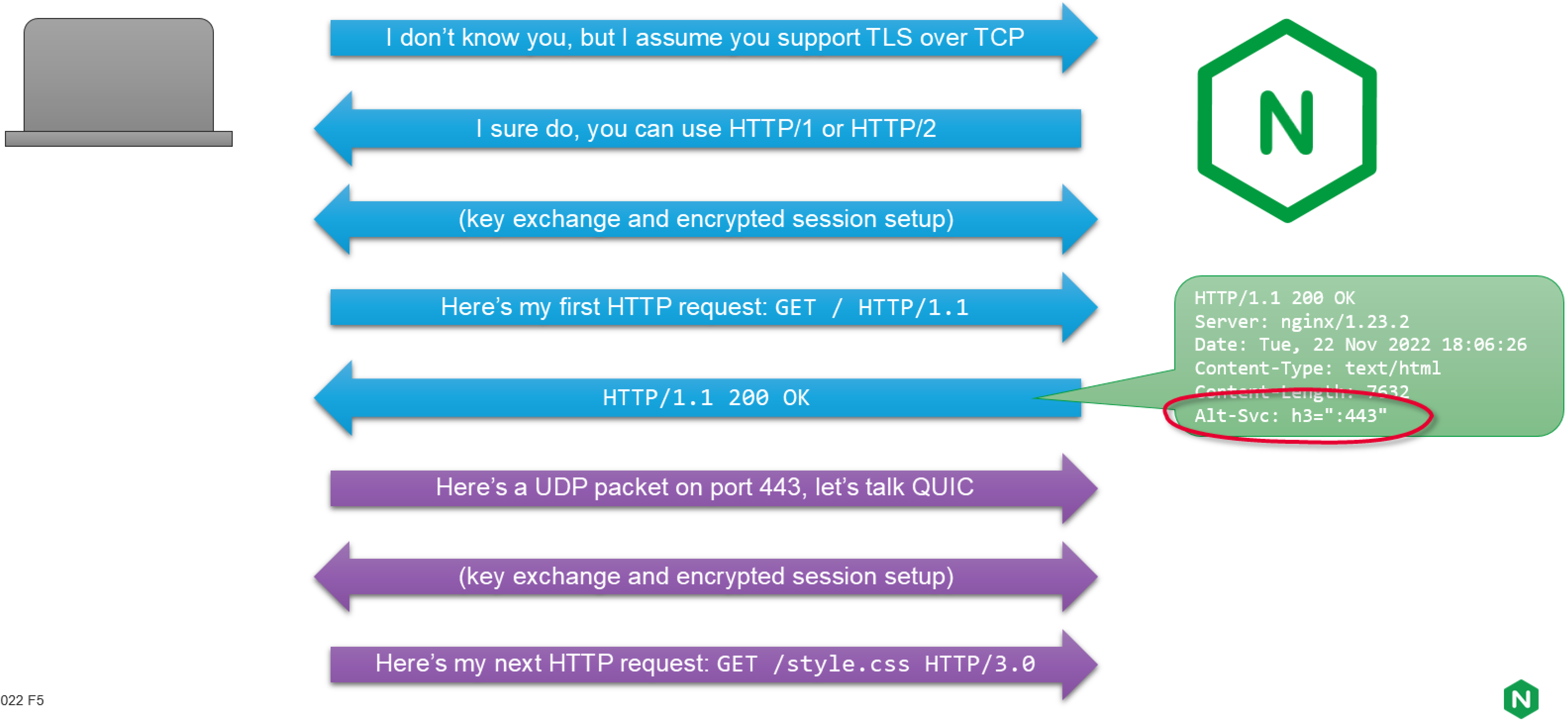
- HTTP to HTTPS: 3xx redirect, Meta, JavaScript
- HTTP to HTTPS: HSTS headers
- HTTP(s)/1 to Websocket: Upgrade header
- HTTP/1 to HTTP/2: Upgrade header, NPN & ALPN via TLS
- HTTP/{1,2} to HTTP/3: Alt-Svc header

Alt-Svc examples

```
Alt-Svc: h2="new.example.com:443"; ma=86400;
Alt-Svc: h3="newest.example.com:50781"; ma=86400;
Alt-Svc: h3=":50781"; ma=86400;
```

Servers MAY serve HTTP/3 on any UDP port, since an alternative always includes an explicit port.

HTTP/3 version negotiation



I don’t know you, but I assume you support TLS over TCP

I sure do, you can use HTTP/1 or HTTP/2

(key exchange and encrypted session setup)

Here’s my first HTTP request: `GET / HTTP/1.1`

`HTTP/1.1 200 OK`

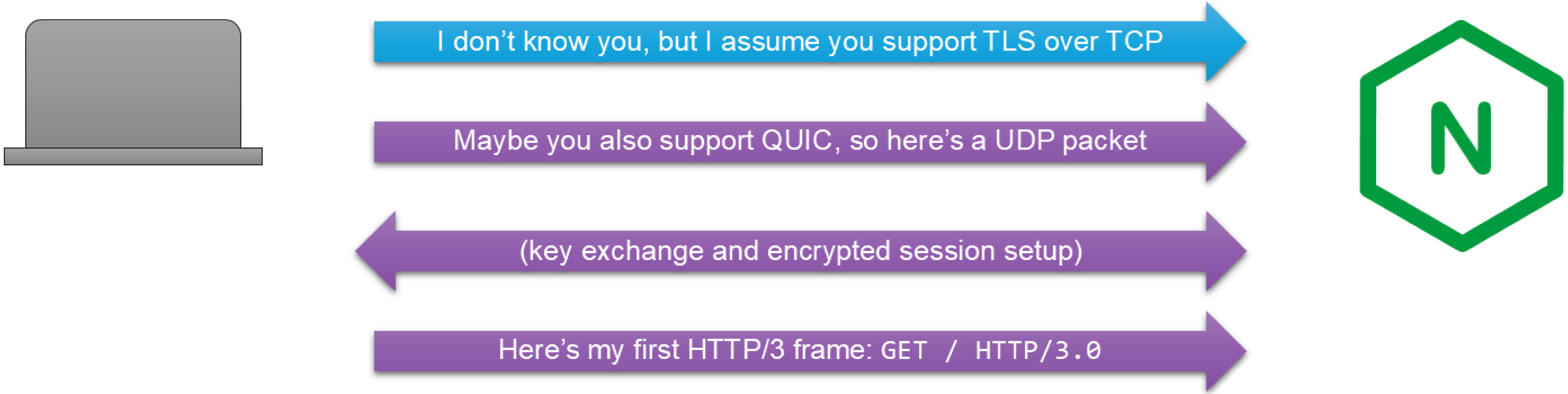
```
HTTP/1.1 200 OK
Server: nginx/1.23.2
Date: Tue, 22 Nov 2022 18:06:26
Content-Type: text/html
Content-Length: 7632
Alt-Svc: h3=":443"
```

Here’s a UDP packet on port 443, let’s talk QUIC

(key exchange and encrypted session setup)

Here’s my next HTTP request: `GET /style.css HTTP/3.0`

HTTP/3 optimistic negotiation



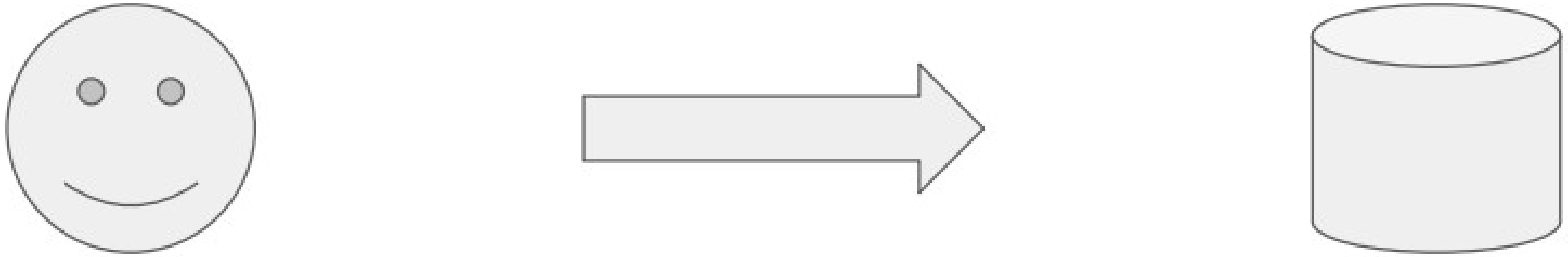
I don’t know you, but I assume you support TLS over TCP

Maybe you also support QUIC, so here’s a UDP packet

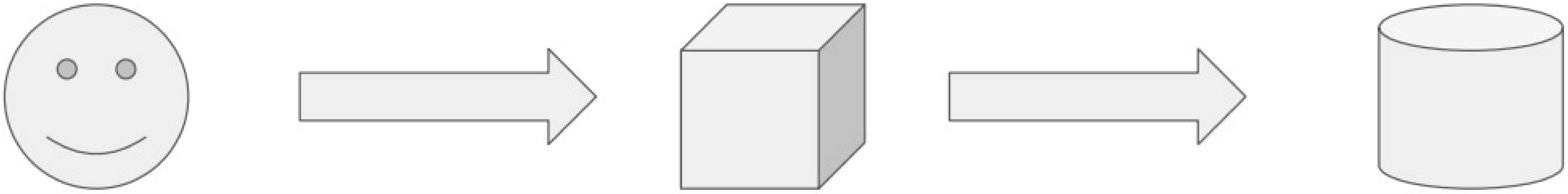
(key exchange and encrypted session setup)

Here’s my first HTTP/3 frame: `GET / HTTP/3.0`

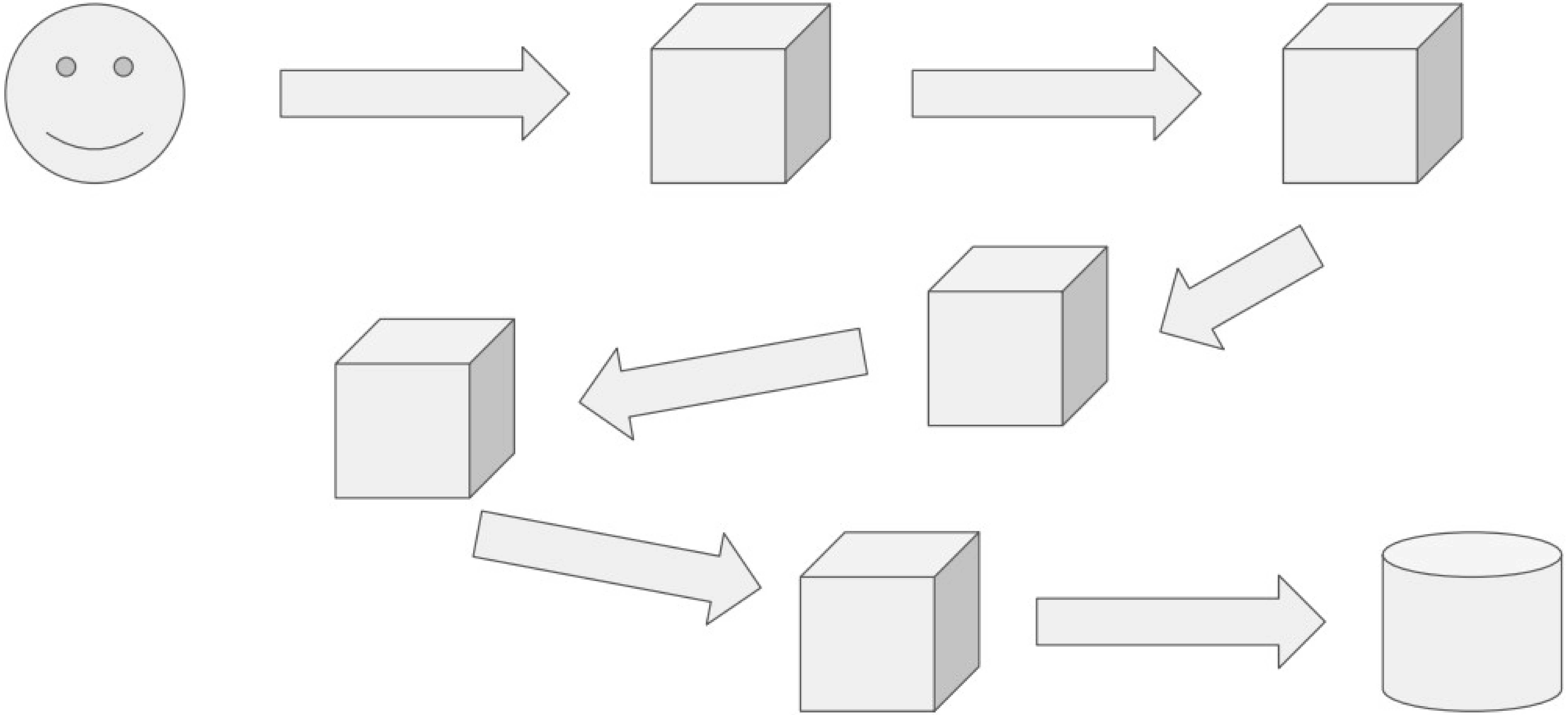
Is this the real world?



Real world is more like this



But really, this



Infrastructure Challenges

- Hardware is tuned for old protocols
- Slow upgrade cycles
- Boxes are not yours
- Requires significant effort between major Internet entities

Server Engineer Challenges

- UDP stack is not optimized
- Need to reimplement features of TCP
- Complicated multiprocessing

Tooling Challenges

- No plaintext version
- Minimal debug tools
- No visibility / monitoring

Security Challenges

- UDP is not trusted due to lots of recent "misuse“
- 0-RTT replay and misconfiguration
- Need to design new security devices
- Conspiracy theories: Google owns both ends of HTTP/3
- Agility of the protocol
  - Overheard yesterday: I’m a bit chaotic, but let’s say “agile”

HTTP/3 with NGINX

- Today: separate branch
  - Howto at quic.nginx.org
- Soon: in mainline

NGINX configuration: HTTP/1 (with TLS)

/etc/nginx/conf.d./proxy.conf

```
server {
    listen 443 ssl;             # TCP listener for HTTP/1


    ssl_protocols       TLSv1.2 TLSv1.3;
    ssl_certificate     ssl/www.example.com.crt;
    ssl_certificate_key ssl/www.example.com.key;

    proxy_pass http://my_backend;


}
```

NGINX configuration: HTTP/2

/etc/nginx/conf.d./proxy.conf

```
server {
    listen 443 ssl http2;       # TCP listener for HTTP/1 and HTTP/2


    ssl_protocols       TLSv1.2 TLSv1.3;
    ssl_certificate     ssl/www.example.com.crt;
    ssl_certificate_key ssl/www.example.com.key;

    proxy_pass http://my_backend;


}
```

NGINX configuration: HTTP/3

/etc/nginx/conf.d./proxy.conf

```
server {
    listen 443 ssl http2;        # TCP listener for HTTP/1 and HTTP/2
    listen 443 http3 reuseport; # UDP listener for HTTP/3 over QUIC

    ssl_protocols       TLSv1.2 TLSv1.3;
    ssl_certificate     ssl/www.example.com.crt;
    ssl_certificate_key ssl/www.example.com.key;

    proxy_pass http://my_backend;

    add_header Alt-Svc 'h3=":$server_port"'; # Advertise HTTP/3 is
available
}
```

Summary

HTTP/1 is not going away

(ever!)

Still well-suited for backends and application runtime

HTTP/2 is already the standard for internet-facing web services

but it failed to deliver on its promises,

and we’re still fixing it!

QUIC+HTTP/3 addresses many of HTTP/2 challenges

Start testing now,

but expect to be Internet-facing-only for some time

Must read

- daniel.haxx.se/http3-explained/

Leave your feedback!

You can rate the talk and give a feedback on what you've liked or what could be improved

++ High Load Armenia

Co-organizer

Yandex